Flexible Automation: Simulation

Adedamola Sode

December, 2021

UNIVERSITÀ DEGLI STUDI
DI GENOVA

Abstract

This project/assignment required the use of CoppeliaSim and the programming language Lua, with a focus on single thread programming to achieve a given industrial scenario: 'A car engine parts manufacturer approaches you seeking a robotic automated solution for one of their problems of loading and unloading, the station receives three parts of the engine, your task is to design a flexible manufacturing cell with robot centred layout for the requirements. The task was implemented successfully with the conveyor spawning various parts randomly and the robot accomplishing pick and place tasks for each part, designating a defined dropzone for each part.

1 Introduction

Coppeliasim is a software formerly known as V-REP for robot simulation used in the industry, educationally and for research. Originally developed by the research and development team of Toshiba, it is now managed by Coppelia Robotics AG. For this semester we aim to learn the use of this software in conjunction with flexible automation principles, course code: 66004, at the University of Genoa. The task at hand requires the implementation of a robot centered work cell, with the aim of facilitating pick and placing operations of three different parts (car engine parts), from a main conveyor belt to three distinct conveyors for each part. The requirements for this simulation:

- Ability to supply multiple vendors at the same time
- Fully Automated solution from product in to out
- Flexible in terms of volume and mix product handling
- Zero-defect product.

2 Part Layout Design and Simulation

2.1 Part 1: Import the part models provided as mesh

The parts given were the oil tray, fuel pump and the camshaft. These parts were imported as merely aesthetic meshes dependent on simple object shaped parents. Each part was imported as shown in figures 1,2, 3, where each dummy is placed in an appropriate location to be picked by the robot, with respect to the part and the world.

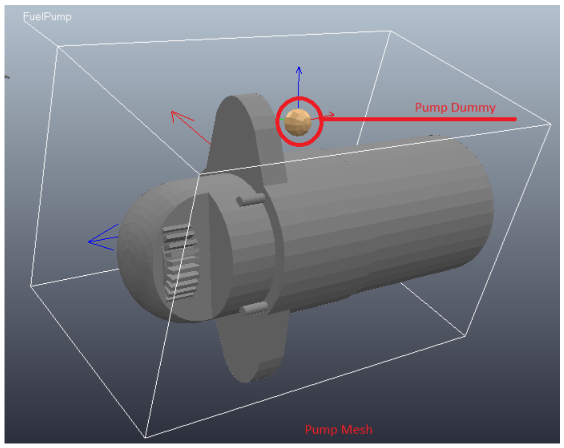


Figure 1: Fuel Pump

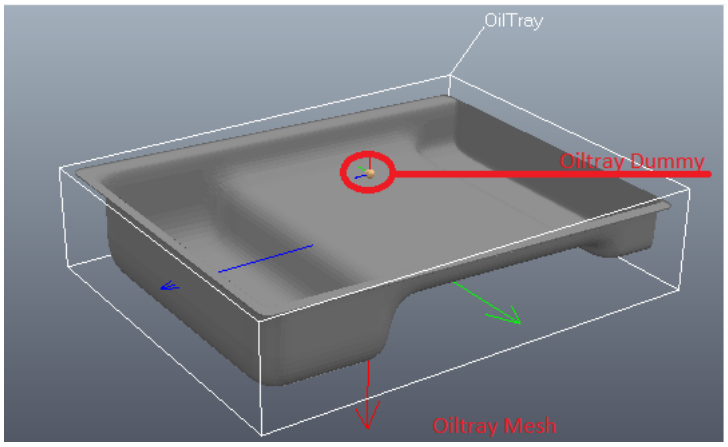


Figure 2: Oil Tray

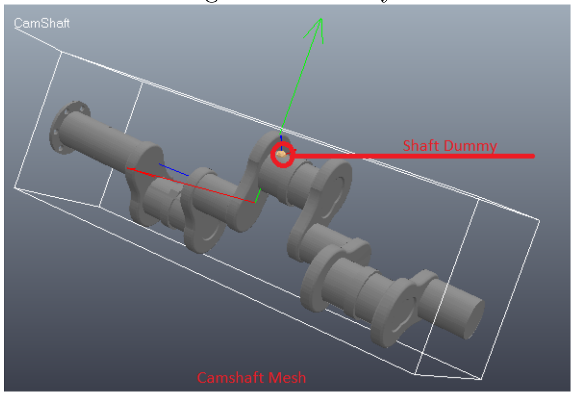


Figure 3: Camshaft

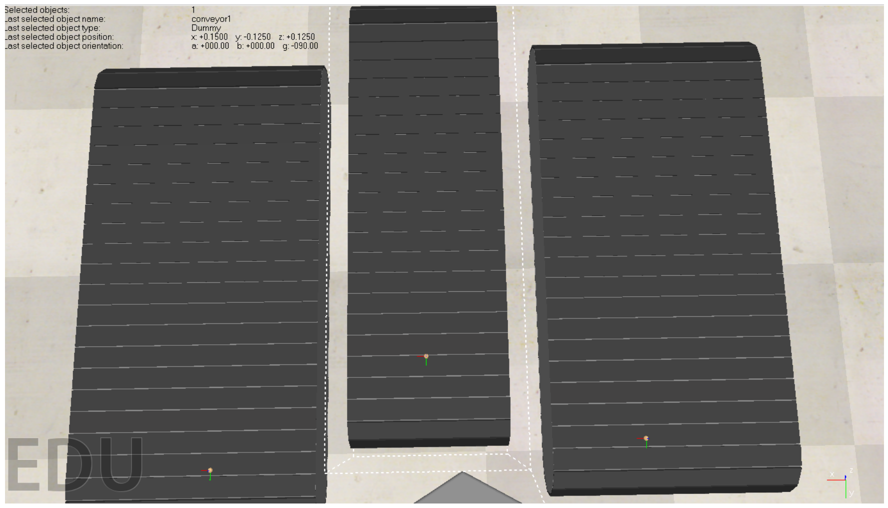
2.2 Part 2: Based on the size of the parts, set up conveyors.

The generic conveyor model was used from the Coppeliasim simulator, adjusting the "Padsize" parameter from the customization script into adapted sizes for each of the parts based on their sizes. The following sizes were utilized:

- **Incoming General Conveyor:** config.padSize= [0.05*2,0.2*2.5,0.005*2]
- **Camshaft Conveyor:** config.padSize= [0.05,0.2*3,0.005]
- **Pump Conveyor:** config.padSize= [0.05,0.2*2,0.005]
- **Oiltray Conveyor:** config.padSize= [0.05,0.2*3,0.005]

2.3 Part 3: Programming Conveyor Motion and Part Spawning

The conveyor movement and part spawning were coded together in a child-script held by the proximity sensor which childs the Incoming general conveyor model. The code required is included with the scene, in the **ProximitySensor** child-script.



```
Selected objects:                      1
Last selected object name:             conveyor1
Last selected object type:             Dummy
Last selected object position:         x: +0.1500   y: -0.1250   z: +0.1250
Last selected object orientation:      a: +000.00   b: +000.00   g: -090.00
```

Figure 4: Conveyors

3 Robot Simulation

3.1 Robot Modeling

The robot modelling aspect of the project met with various issues regarding debugging and model behavior. A RRR robot was initially implemented, where reach and dexterity were considered, but the model reacted extremely buggy and unrespondable, due to my little experience with the software, I was unable to determine the issues and switched to a PRRP robot, as a PRR robot is perfect for pick and place tasks, with a robot centered cell. The final prismatic joint was added to enable better reach for the robot. The CAD files for the first robot implementation were made myself, but the second was sourced from previous class exercises, as they merely serve aesthetic objectives, with not much time on my hands I have chosen to forsake redoing them, but will include the two robots as a review of my total work implemented.

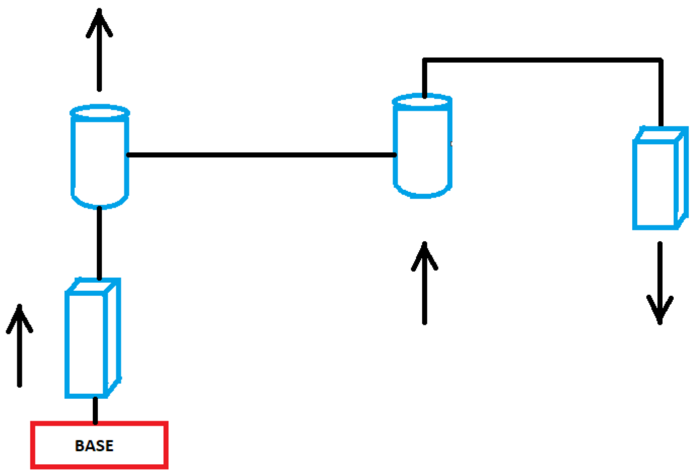
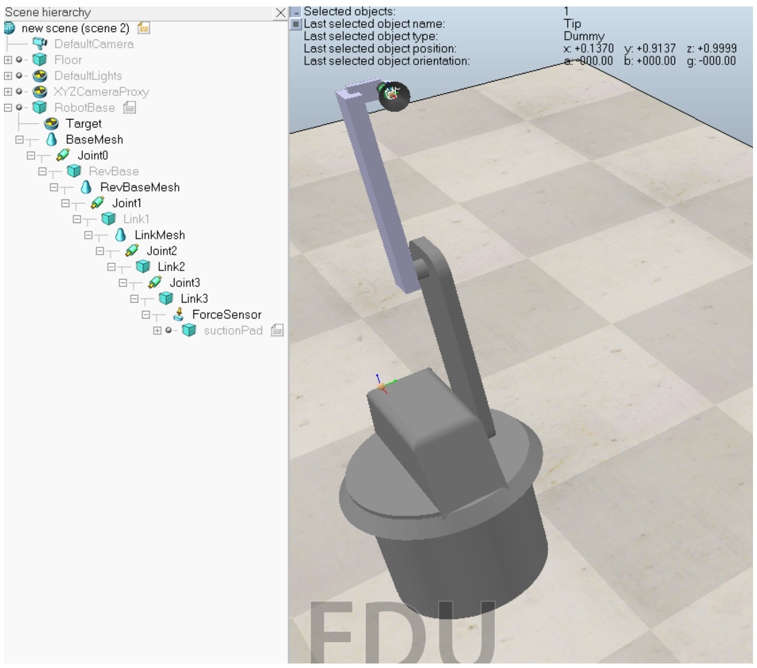


Figure 5: PRRP Robot Schematic

The Schematic diagram provided shows joint combination of the implemented robot, which consists of 2 prismatic joints at each end and two revolute joints in-between.



```
Selected objects:                  1
Last selected object name:         Tip
Last selected object type:         Dummy
Last selected object position:     x: +0.1370   y: +0.9137   z: +0.9999
Last selected object orientation:  a: -000.00   b: +000.00   g: -000.00
```

Figure 6: RRR Robot

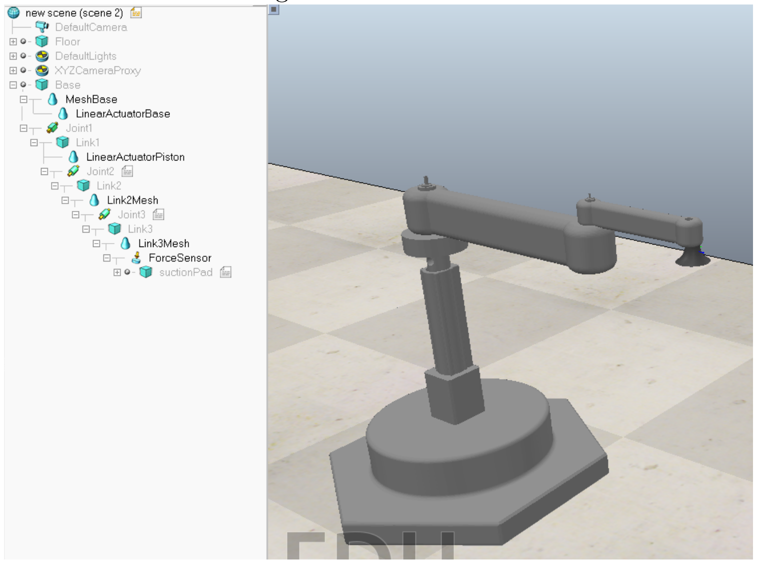


Figure 7: PRRP Robot

For each robot - with my expertise with the software as a factor for my consecutive choices - for simplicity, the suction pad gripper was selected, as the implementation for executing the picking and dropping logic is quite simple.

3.2 Kinematic Modeling

The kinematics of the robot was implemented according to the lectures. For the pick and place logic, and general behavior, the non-threaded code is implemented in the **ProximitySensor** child-script.

3.3 Behaviour Logic

The code works as described by the comments in the scripts. Algorithmically, these steps are attempted and concluded for the scene to be considered a success:

1. Object handlers of the scene players are obtained.
2. Function to spawn parts is implemented from an example given by the professor, with edits made to execute the code based on an indicator that varies from 1 to 3, based on which part is being deployed.
3. When parts are spawned, they are automatically grouped using the GroupParts() function, which creates a table for each of the three possible cases: Fuel Pump, Oil Tray, Camshaft.
4. If the object makes its way to the proximity sensor located midway on the general conveyor, the conveyor stops to enable the robot successfully pick the object.
5. Picking the object is as simple as setting the Target dummy from the Inverse-Kinematics setup at the object position when it reaches the proximity sensor, and when the object attaches, setting the Target dummy to a dummy attached at the 3 remaining conveyors.
6. Based on the object identity, it chooses which final destination dummy to set as the final drop location, and using the Inverse-Kinematic functions setup, proceed to move the robot to that destination
7. Once at the final position, the gripper drops the object by removing the dummy attachment from the object and back to the gripper.
8. The process is repeated for each random spawn

The final scene is made to look like this:

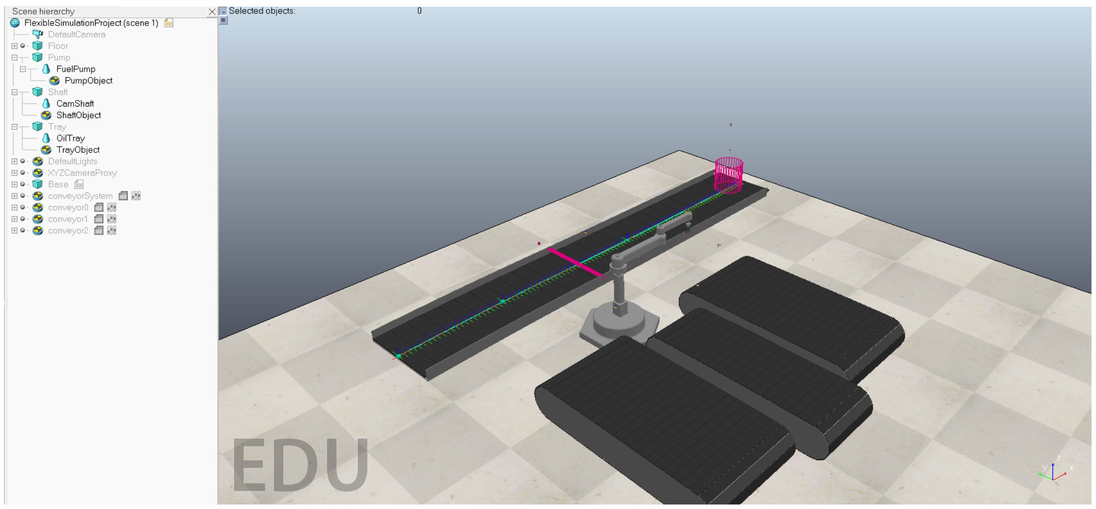


Figure 8: Functional Scene

4 Conclusion

The implementation of the robot for pick and place purposes was a success. The software was not too difficult to use, but consisted of many fine tuning elements that greatly delayed and influenced final implementation. These issues have been mostly sorted to accomplish the main objective of the project, but clearly show a big disconnect between my knowledge/experience with the software and the required tasks. There are more things to be done regarding the project to fine tune performance and general scene-player behaviours, but in requirement of the course objective, final submission is necessary.